Intrusion detection by a dynamic length rule

Al Naqshbandi, S.M.; Samawi, V.W.; van den Herik, H.J.

Published in:
[n.n.]

Publication date:
2012

Document Version
Publisher's PDF, also known as Version of record

Procedia Information Technology & Computer Science

00 (2013) 000-000

3rd World Conference on Information Technology 2012

Intrusion Detection by a Dynamic Length Rule

Susan M. Al Naqshbandi [a] *, Venus W. Samawi [b], Jaap van den Herik [a]

[a]Tilburg University, P. O. BOX 90153, 5000LE, The Netherlands
[b]Al al-Bayt University, P. O. BOX 130040, 25113 Mafraq, Jordan

Abstract

Intrusion Detection Systems (IDSs) are used to establish if someone has made an intrusion into the network or is trying to make one. Many techniques are available to construct IDS using genetic algorithm. But, all are based on a fixed length rule. In this paper, we propose to improve the IDSs by using dynamic length rule with an automatic feature selection. The proposed improvement accounts for the complexity of the data by using two of the most popular methods of soft computing, namely Fuzzy Logic and Genetic Algorithm. For a proper determination of the rule length we apply iterative rule learning, based on a fuzzy rule-based genetic classifier. We distinguish five main classes, viz. Normal, User-to-Root (U2R), Probe, Remote-to-Local (R2L), and Denial-of-Service (DoS). The First aim of the paper is to suggest an automatic method for producing rules (chromosomes) of dynamic length. The chromosome length represents number of features involved in the corresponding rule. The second aim is to evolve comprehensible rules that improve the classification rate for each of the five classes. In the paper the performance of the evolved rules is given per class. The obtained results provide the detection rate with regards to the lowest number of features for each class.

Keywords: Intrusion, Fuzzy Logic, Genetic Algorithms;

Selection and/or peer review under responsibility of Prof. Dr. Dogan Ibrahim.

1. Introduction

The explosion of internet applications has resulted in more attention for safety and security. The main question still is: how to protect the stored information adequately? Clearly, traditional network detection has failed to protect networks completely against the ingenious attacks that have been developed in the last decade. As a result, many attempts to develop dedicated network intrusion detection systems have been performed to detect intrusions. An intrusion is defined as any set of actions that attempts to compromise the integrity, confidentiality, or availability of a resource (Heady et al., 1990). Two main approaches have been devised to detect intrusion: (1) misuse detection and (2)

* Susan, Al Naqshbandi, Tilburg University, P. O. BOX 90153, 5000LE, The Netherlands
E-mail address: dr.susan_alnaqshbandi@yahoo.com / Tel.: +31-064-144-0949

anomaly detection. Misuse detection searches for specific patterns or sequences and user behaviour that match well known intrusion scenarios. Anomaly detection develops models of normal network behaviours; new intrusions are then detected by evaluating significant deviations from the normal behaviour. There are many approaches that propose a model to handle the anomaly-based intrusion detection [(Tajbakhsh et al., 2009), (Wu, 2010), and (Deepa, & Kavitha, 2012)]. Among them we see statistical techniques (Asaka et al., 2001), machine learning (Li, & Guo, 2007), data mining (Nguyen, & Choi, 2008), soft computing (Chou et al., 2008), and immunological inspired techniques (Powers, & He, 2008). For historical reason we mention Denning (Denning, 1987) who was the first to propose an intrusion detection model which described several statistical techniques for an anomaly detection system including threshold measure, mean, standard deviation, and multivariate models. Jumping to the modern time we mention the advanced computational intelligence algorithms including genetic algorithms (GA). Moreover genetic programming (GP) is reported in (Wu, & Banzhaf, 2012) as a mean for solving IDS problems. A brief historical overview is given below.

Around 1990 no fewer than 41 features (taken from the KDDcup dataset) were established to be checked by a detection rule for prohibiting an intrusion. Of course, these 41 features took a considerable amount of time and resources. Therefore, it was investigated whether an IDS could do the same job with a fewer number of features. This is a delicate question since it is related to the nature of an attack which in turn, is closely related to the input for the system to be attacked. We consider the five main classes of input, viz. Normal, User-to-Root (U2R), Probe, Remote-to-Local (R2L), and Denial-of-Service (DoS). If Normal input is expected then we check by three features (our assumption) whether the input is really normal. If it is not normal then we do not allow the input to enter the system. We note that classical inputs with hidden tricks are traced by applying more features. However, that is not the way we wish to follow. In our proposed system we conjecture that at most 20 features are sufficient for detecting the most heavy method of intruding the system namely by DoS. In the paper we will investigate the number of features necessary for appropriate IDSs for all five classes.

For this purpose, we consider two popular methods of soft computing: (1) Fuzzy Logic (FL) and (2) Genetic Algorithm (GA) for selecting a subset of significant features from a feature set of network data. The reason behind using FL is: by the involvement of many quantitative features that suffer from vagueness and imprecision (i.e., there is no clear separation between normal operations and anomalies). Fuzzy rules can be useful to discriminate such data type (Lee, 1990). Different techniques have been suggested for automatic generation and modification of fuzzy rules without using the aid of human experts. On the other hand, GA is one of the most successful approaches for the appropriate generation of chromosomes followed by a suitable modification [(Gomez, & Dasgupta, 2002), (Abadeh et al., 2005), (Tsang et al., 2007), (Abadeh et al., 2008), and (Victoire, & Sakthivel, 2011)]. GA uses iterative learning approach. The population is selected by a “guided” random search using parallel processing to achieve the desired result. To specify the best feature set that could be used to recognize if the connection is an intrusion or not, we suggest dynamic chromosome size to be used. So far, all researchers used a static chromosome size for a pre-selected subset of features. But we believe that using a dynamic chromosome size is a better approach. The idea is as follows. The chromosome length is set randomly in advance, and the chromosome is also filled randomly. Both, the length and the contents are to be adapted by a learning process as will be illustrated later.

In summary, our idea is to present the applicability of genetic algorithms to evolve a set of fuzzy rules that can solve some well-studied intrusion detection problems according to the two methods.

The development of our idea is as follows. First, we improve the ideas of Gomez and Dasgupta (Gomez, & Dasgupta, 2002) by using a dynamic length rule (each rule is represented as chromosome). This will give us a more accurate detection rate. Second, we evaluate the performance of our improved genetic-fuzzy classifier using the reduced KDDCup dataset (i.e., we input variables given by Mukkamala, & Sung, 2003).

The remainder of the paper is organized as follows. Our new system is proposed in Section 2. Experiments are described in Section 3. Section 4 provides the results. In Section 5 we present our conclusions.

2. Proposed system

We propose an enhanced attack-detection model using a revised One-Rule Genetic-Fuzzy Classifier (Al Naqshbandi, & Samawi, 2012). Our system is basically a fuzzy system augmented by a learning process based on GA. The main steps for implementing the proposed system are illustrated in figure (1). At first, the dataset (taken from KDDCup (KDD-cup dataset, 2012)) is preprocessed (discussed in 2.1), then split into two datasets (training set and test set). GA is applied on the specified training dataset to evolve an efficient rule. When a rule is generated, it is applied on the test set to check the rule detection ability. Here, we keep in mind that our aim was to evolve one rule of variable length for each main class (Normal, U2R, Probe, R2L, and DoS). U2R, Probe, R2L, and DoS are the main attack classes. The training phase will be discussed in section (2.2), and the test phase will be discussed in section (2.3).

2.1. Preprocessing

We use the KDDCup 1999 (KDD'99) intrusion detection contest data which is a subset of DARPA dataset. The KDD'99 is considered to be an adequate dataset that has been found to have the required potential for modeling the attacks that appear commonly in the network traffic. Hence, this dataset can be considered as the base line of any research [(Thomas et al., 2008), (Tavallaee et al., 2009), and (Tavallaee et al., 2010)].

The KDD'99 dataset contains normal data and four general attack data types. There are 41 features in this dataset (i.e., N=41). The majority of features have a different scale. For instance, the destination host count is in the range 0 to 255, and the source bytes range from 0 to 693,375,640. Thus, any type of anomaly analysis requires a normalization of the employed features. In order to avoid biases as much as possible we applied data normalization over the original KDD'99 dataset. The goal of data normalization is to ensure a common ground for the subsequent direct comparison of the data points. So each numerical value in the dataset is normalized between 0.0 and 1.0 according to equation (1):

$$y = \frac{x - Min}{Max - Min} \qquad (1)$$

where x is the numerical value of the attribute, Min is the minimum value for the attribute that x belongs to, and Max is the maximum value for the attribute that x belongs to.

The regular partitioning of the data into a training phase and test phase is p% and 100-p %. In our case, we use 10 times cross validation in which each set is partitioned into two sets: (1) 90% of the data goes to the training set, and then (2) 10% should go to the test set. Since we would like to know the result of the whole data set and compare the result with other results, we decided at the end to take the whole set as test set. We note that the data splitting was done after the data normalization.

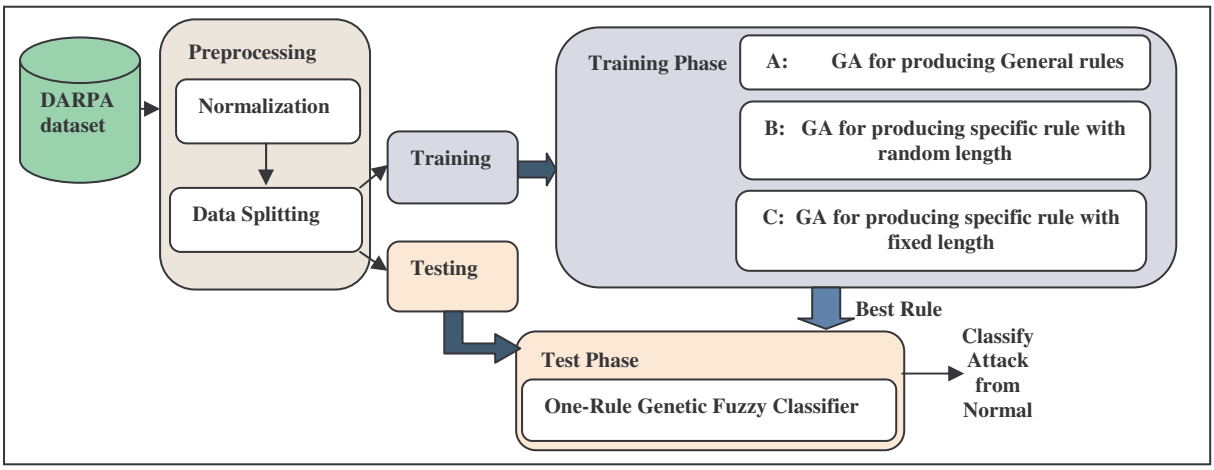

Figure 1. The Block Diagram of the proposed system

2.2. Training Phase

The training phase is the most important phase. In this phase the high-detection-rate rule with a suitable number of features will be evolved by one of the procedures A, B, or C (see figure 1 and Section 3). Below we discuss the training data (2.2.1), the Fuzzy Logic Rules (2.2.2), the Chromosome Encoding (2.2.3), Two Rule types (2.2.4), and the Evolving of Fuzzy Rules by a Genetic Algorithm (2.2.5).

2.2.1 Training data

The training data can be handled in two ways: (1) use a general data splitting method, and (2) use a data filtering splitting method. Both methods have their cons and pros. In our experiment we have investigated both methods, we describe them briefly below.

- **General data splitting method:** The whole training set is split into two types: (1) Normal connection activities, and (2) Attack connection activities (with different types of attack). To build the training datasets, three sorts of random selections for connections are used ((1) *approximately equal number of activities from both types*, (2) *equal percentage of activities from both types*, or (3) *different percentage of activities for each type*).

- **Data filtering splitting method:** The whole training set is divided into five types: (1) Normal data type, (2) U2R attack type, (3) Probing attack type, (4) R2L attack type, and (5) DoS attack types. To build the training datasets, two forms of random selection of connections were used (*equal percentage of activities from both types* i.e., Normal and the considered attack, or *different percentage of activities for each type*).

2.2.2 Fuzzy Logic Rules

Fuzzy Logic (FL) is a set of concepts and approaches designed to handle vagueness and imprecision. A set of rules can be created to describe a relationship between the input variables and the output variables, which may indicate whether an intrusion has occurred (Su et al., 2008)]. In fuzzy logic, (1) fuzzy sets define the linguistic notions and (2) membership functions define the truth-value of such

linguistic expressions (Al Naqshbandi, & Samawi, 2012). By taking fuzzy spaces into account, fuzzy logic allows an object to belong to different classes at the same time. The idea of belonging to more than one class is helpful when the difference between classes is not well defined. It happens frequently in the intrusion detection task, where the difference between the normal and abnormal class are not well defined.

A **fuzzy rule** is defined as a normal conditional statement in the form: **IF** *x is A* **THEN** *y is B*

where x and y are normal linguistic variables; the essence is that A and B are linguistic values determined by fuzzy sets and spaces.

The membership degree to a fuzzy set is called the membership function. In this work the triangular membership function is used. For non-numerical attributes we used the categorical values as crisp sets (fuzzy sets that does not overlap each other). For each numerical attribute, we assign the following fuzzy space (i.e., collection of fuzzy sets that define the fuzzy linguistic values to which an object can belong) Low= {0.000-0.333}; Medium Low= {0.166-0.500}; Medium= {0.333-0.666}; Medium High= {0.500-0.833}; High = {0.666-1.000}.

2.2.3 Chromosome Encoding (Rule Representation).

We assume that the reader has some knowledge of GA [(Melanie, 1999) and (Haupt, & Haupt, 2004)]. The encoding schema used in our work is direct value encoding (we refer to problems, where some difficult values, such as real numbers, are used) (Al Naqshbandi, & Samawi, 2012). Every chromosome is a string of values. Values can be anything that is connected to the problem, e.g., integer numbers, real numbers and characters connected to some complex objects. Since different GA is trained for each class, the action part does not need to be represented as part of the rule in the chromosome. The rule (chromosome) only represents the condition part. Formally, a condition is generated by a grammar such as the one described in (Al Naqshbandi, & Samawi, 2012).

As will be illustrated in figure 2, a chromosome is defined as a set of *n* genes (ranging from 0 to n-1; we recall that the number of genes is N=41). An atomic condition and a fuzzy operation compose a gene. Each atomic condition contains Feature number, variable, relational operator, and the set. Fuzzy operation contains two fields: (1) fuzzy operator and (2) its precedence number. However, there is one exception: the last gene, is composed of an atomic condition only, in that gene the last part (Fuzzy Operator) is ignored. Assume that the Normal class has three genes (e.g., 2, 3, 6 then n=3 and Fno_i should read Fno_2, Fno_j should read Fno_3, and Fno_k should read Fno_6).

In this work, each chromosome has a structure record of length ***>2 and <=41***. An important characteristic of this representation is that, in order to express the genotype, the classical parsing algorithm, *operator precedence parser* (Aho et al., 2001), is used. This technique allows the evaluation of an expression in an efficient way. The chromosome has to be traversed only once, that is, the time complexity of the evaluation is *O (n)* (with *n* being the condition expression length).

Gene $_0$						...	**Gene $_{n-2}$**						**Gene $_{n-1}$**				
Atomic condition$_i$				**Operation$_i$**		...	**Atomic condition$_j$**				**Operation$_j$**		**Atomic condition $_k$**				*
Fno_i	**var_i**	**ro_i**	**set_i**	**op_i**	**$prec_i$**	...	**Fno_j**	**var_j**	**ro_j**	**set_j**	**op_j**	**$prec_j$**	**Fno_k**	**var_k**	**ro_k**	**set_k**	*

Figure 2. Chromosome representation of the condition

2.2.4 Two Rule Types

To identify the best feature set that could be used to recognize whether the access activity is an intrusion or not, we propose two types of rules. The chromosome length has (1) a random number of genes, and (2) a fixed number of genes.

- **Random number of genes:** we propose a dynamic rule length. All former researchers used static chromosome size for a pre-selected subset of features. We believe that a dynamic chromosome size (at which each chromosome length is specified randomly, and then the chromosome is filled also randomly) will give the flexibility to choose the suitable features that relate to the problem in hand according to its fitness value. As well known, extra useless features may degrade the classification ability, so it is better not to use them. Dynamic chromosome length will help in disposing useless features. A population of chromosomes with a variable length is created randomly. Each gene-field in the chromosome (see figure 2) will be filled randomly (according to its proper range) except the *var* field because when the rule will be applied over the input activity it will take that variable from the activity connection to compute the result.

- **Fixed number of genes:** A population of a fixed number of genes is created according to the idea of performing experiments to rank the importance of the input features for each of the five classes (Normal, U2R, Probe, R2L, and DoS) of the patterns in the KDD'99 data. We adopted the idea that using only the important features for classification will gives good accuracies and, in certain cases, will reduce the training time and test time of the classifier (Mukkamala, & Sung, 2003). Using feature ranking (a fixed number of genes), Mukkamala and Sung arrived at the following fixed features for each class. The numbers are the feature number in the specified dataset, where they range from 1 to 41.

  Normal = 25 = (1,3,5,6,8,9,10,14,15,17,20,21,22,23,25,26,27,28,29,33,35,36,38,39,41),
  U2R = 8= (5,6,15,16,18,25,32,33),
  Probe =7= (3,5,6,23,24,32,33),
  R2L=6= (3,5,6,24,32,33).
  DoS= 20= (1,3,5,6,8,19,23,24,25,26,27,28,32,33,35,36,38,39,40,41),

2.2.5 Evolving Fuzzy Rules by a Genetic Algorithm

GA mimics the natural reproduction system in nature where only the fittest individuals in a generation will be reproduced in subsequent generations, after undergoing recombination and random change (Owais et al., 2008). Below we discuss the three main steps of the genetic algorithm that we implemented in the proposed system viz. population generation, fitness evaluation, and genetic operators.

- **Population Generation**: The population undergoes the evolution in the form of natural selection. The successive iterations are called generations. In this step a set of *200* chromosomes (rules) is randomly generated consisting of *n* genes, each of which contains the fields described earlier. See Figure 3.

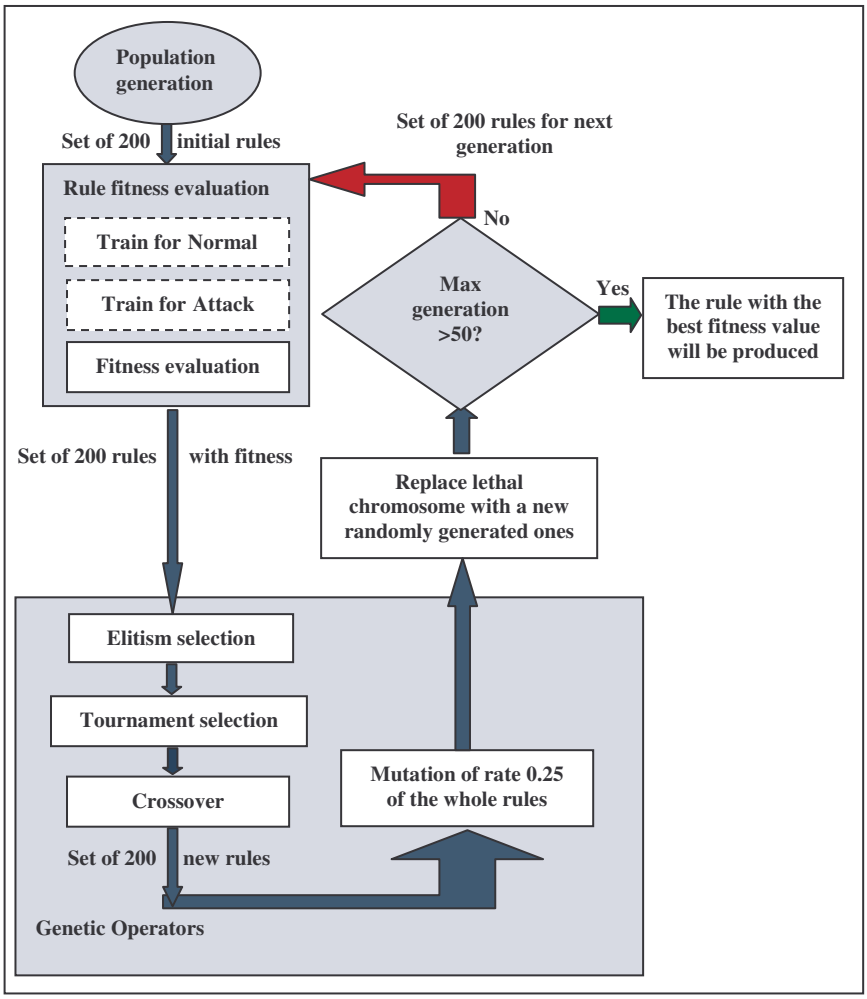

Figure 3. The flow chart of our GA

- **Fitness Evaluation: The** GA processes populations of chromosomes (individuals) as follows. The algorithm successively replaces one such population by another population depending on the fitness function. The fitness function assigns a score (fitness) to each chromosome in the current population. The fitness of a chromosome depends on how well that chromosome solves the problem at hand. Given a particular chromosome, i.e., a possible solution, the fitness function returns a single numerical value, which is supposed to be proportional to the utility or adaptation of the solution represented by that chromosome. The fitness of each chromosome (rule) is evaluated with respect to a set of attribute vectors (training set) to which a class has been previously assigned. In each run of the GA, a rule for different class C_i is evolved. The fitness of a chromosome for the normal and attack class is described in (Al Naqshbandi, & Samawi, 2012). See Figure. 3.

- **Genetic Operators**: A genetic operator is a process used in GAs to maintain an appropriate genetic variation. Genetic variation is a necessity for the process of evolution. Genetic operators used in GAs are analogous to those which occur in the natural world. They are: (1) survival of the fittest, or selection, (2) reproduction (crossover, also called recombination), and (3) mutation (Holland, 1975). Chromosomes in the population are rated for their adaptation to a basis of the ratings; a new population of chromosomes is formed using a selection mechanism and specific

genetic operators such as crossover and mutation. In this work, two type of selection are used: (1) Elitism, and (2) Tournament. The genetic classifier program is trained so many times depending on (1) which data to be classified, (2) which type of rule (random number of genes or fixed number of genes), and which rule (Normal or attack).

2.3. Test Phase

In the testing phase, one of the rules generated in the training phase is used to classify the normality or abnormality of a specific pattern. The input to the classifier is (1) the rule and (2) the pattern to be classified. The process starts by fuzzifying the pattern according to the rule, then calculating the predicted value (by the operator precedence parser (Aho et al., 2001)). The result is one value. According to that value and a specific threshold, a decision can be made (the pattern is either normal or a specific attack).

3. Experiments

In the experiments, we investigate how the correlation between features might influence the classification result. Accidental elimination of important features might decrease the classification accuracy. Furthermore, some features might have no (or sometimes negative) effect. The removal of such features can increase the search speed and the accuracy rate.

Classification is used to find a logical description that correctly classifies the novel cases. The rules that are generated from training the genetic algorithm need to be tested to measure the ability of each rule in classifying its corresponding attack. So, to study the behaviour of the classifier (rule), the whole filtered dataset is used for measuring the classification efficiency of the generated rules using (2).

$$C = \frac{S}{T} \qquad (2)$$

where C is the classification efficiency, S is the number of correctly classified activities, and T is the total number of the activity records in the dataset.

The main classification idea used in this work depends on the generated rule and the threshold. Selecting a threshold value is necessary to help the intrusion detection for making a good decision in identifying the attacker. It is difficult to select a suitable threshold value for differentiating between a normal activity and abnormal activity. Therefore, it is important to find out the proper threshold values (a threshold value for normal activities, a threshold value for the attack activities). Choosing the best threshold values can be done by data observation and a heuristic trial-error method. The best threshold value for a normal activity is 1.0 and the best threshold value for the attack activity is 0.01.

After the training phase, our genetic algorithms (A, B, and C; see figure. 1) produced 5 promising rules. The production was as follows. A produced R1; B produced R2, R3, and R4; C produced R5. The specification reads:

A: (R1) using general splitting method and random number of genes with 54615 normal activities we obtained a rule that can classify Normal activities.

B: (R2) using data filtering splitting method and random number of genes with 230 U2R attack type we obtained a rule that can classify U2R attack type.

(R3) using data filtering splitting method and random number of genes with 4166 Probe attack type we obtained a rule that can classify Probe attack type.

(R4) using data filtering splitting method and random number of genes with 8129 R2L attack type we obtained a rule that can classify R2L attack type.

C: (R5) using data filtering splitting method and fixed number of genes with 114935 Dos attack type we obtained a rule that can classify Dos attack type.

4. Results

Below we show the features used in the five rules R1, R2, R3, R4 and R5. The numbers represent the feature numbers in the KDD'99 dataset, where they range from 1 to 41. For comparison with (Mukkamala, & Sung, 2003) we refer to subsection (2.2.4).

Normal = 3 = (2, 3, 6),
U2R = 3= (1, 2, 8),
Probe =7= (1, 2, 11, 13, 18, 21, 23),
R2L=13= (2, 6, 12, 14, 15, 16, 17, 18, 24, 30, 31, 33, 38),
DoS= 20= (1,3,5,6,8,19,23,24,25,26,27,28,32,33,35,36,38,39,40,41).

In the test phase we applied the five rules produced during the training phase to the full test dataset. Table 1 illustrates the detecting behavior of the rules.

Table 1. The rules that can classify a specific attack

Rule Number	Detecting type	Number of features	Amount of testing data	Truly detected	Falsely detected	Detection Rate %
R1	Normal	3	60593	60214	379	99.37
R2	U2R	3	230	230	0	100
R3	Probe	7	4166	4027	139	96.66
R4	R2L	13	16187	16187	0	100
R5	DoS	20	229853	229848	5	99.99

Clearly, we reduced the number of features for the Normal class from 25 to 3. Closer inspection shows that the intersection of the two sets consist of the features (3, 6). Feature (2) is not in the (Mukkamala, & Sung, 2003) set. In table 2 we provide a complete overview of the comparison.

5. Conclusion

The main contribution of the present work is (1) the dynamic chromosome length with the application of the two soft-computing methods (Fuzzy Logic and Genetic Algorithms); and (2) achieving a classification model with a high detection accuracy and with a lower number of features. Since, we did not use all features of the set of 41 features. In particular we did not use features (4,7,9,10,20,22,24,34,37). A fair comparison with the Mukkamala is difficult. We were able to reduce the feature set for the normal class and U2R class, but we have to broaden the feature set for the R2L class. Moreover, we were able to achieve a score of 100 % for the classes U2R and R2L, whereas MuKKamala and Sung did not reach the perfect score. More research is needed for a definitive conclusion.

Table 2. Comparison between our work and Mukkamala's work

Rule Number	Detecting type	Number of features for our work	Number of features in Mukkamala	Feature Similarity	Our detection rate %	Mukkamala detection rate %
R1	Normal	3	25	3,6	99.37	99.59
R2	U2R	3	8	No	100	99.87
R3	Probe	7	7	23	96.66	99.38
R4	R2L	13	6	6,24,33	100	99.78
R5	DoS	20	20	same	99.99	99.22

In this work we tried to show a distinct preference for detecting all the classes with an improved accuracy. However, The accuracy of detecting an individual attack and the normal connection is as follows: detecting Normal connection is 99.37%, U2R attack connection 100%, Probe attack connection 96.66%, R2L attack connection is 100%, and DoS attack connection 99.99% (see table 1and 2). This result is to be qualified as *good;* it compares favourably to other approaches reported in the literature. The main difficulty is in Probe. We will investigate how we can improve the detection rate. Remarkably, both approaches use a set of 7 features of which the intersection only contains feature (23). All in all, the obtained results demonstrate the effectiveness and applicability of the proposed method. The combination of the two soft computing methods gives an adequate performance of the IDS, and makes the detection quite effective. The result of the evaluation produced a better result in terms of (1) detection efficiency and (2) using dynamic rule length when applied to the existing problems. In summary, our proposed method presents useful information for a new approach of intrusion detection. Finally, we may claim that our secondary goal has been achieved by producing comprehensible rules. The claim will be substantiated in a follow-up contribution.

References

Abadeh, M.S., Habibi, J. & Lucas, C. (2005) Intrusion detection using a fuzzy genetics-based learning algorithm. *Journal of Network and Computer Applications*, doi:10.1016/j.jnca,05.002.

Abadeh, M. S., Habibi, J. & Soroush, E. (2008, September). Induction of Fuzzy Classification Systems Via Evolutionary Aco-Based Algorithms. *IJSSST*, Vol. 9, No. 3.

Aho, A.V., Sethi, R. & Ullman, J.D. (2001). *Compilers: Principles, Techniques, and Tools*. Addison-Wesley.

Al Naqshbandi, S. M. & Samawi, V. W. (2012). One-Rule Genetic-Fuzzy Classifier. *Proceedings of the Computer Science and Automation Engineering IEEE*, pp:204-208.

Asaka, M., Onabura,T., Inoue,T., Okazawa, S. & Goto,S. (2001, May). A new intrusion detection method based on discriminant analysis, *IEICE Transactions on Information and System* 5, 570–577.

Chou, T. S., Yen, K. K. & Luo, J. (2008). Network Intrusion Detection Design Using Feature Selection of Soft Computing Paradigms,*International Journal of Computational Intelligence* Volume 4 Number 3.

Deepa, A. J. & Kavitha,V. (2012). A comrehensive survey on approaches to Intrusion Detection System, *Elsevier, Procedia Engineering* 38, 2063-2069.

Denning, D.E. (1987, February 2). An intrusion detection model, *IEEE Transactions on Software Engineering* vol. SE-13, 222–232.

Gomez, J. & Dasgupta, D. (2002). Evolving fuzzy classifiers for intrusion detection. *IEEE Workshop on Information Assurance*, United States Military Academy, West Point, NY, USA, pp: 68–75.

Haupt, R. L. & Haupt, S. E. (2004). *Practical Genetic Algorithms*. (2nd ed.), John Wiley & sons, inc.

Heady, R., Luger, G., Maccabe, A., & Servilla, M. (1990, August). The architecture of a network level intrusion detection system. *Technical Report CS90-20, Department of Computer Science, University of New Mexico*. http://cnc.ucr.edu/security/glossary.html

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems. The University of Michigan Ann Arbor*.

KDD-cup dataset. (2012). Retrieved January 15, 2012, from http://kdd.ics.uci.edu/data bases /kddcup99/kddcup.html.

Lee, C.C. (1990). Fuzzy logic in control systems: fuzzy logic controller, Part I and Part II. *IEEE Transactions on Systems, Man, and Cybernetics*, 20 (2), pp: 404-435.

Li, Y. & Guo, L. (2007, December). An active learning based TCM-KNN algorithm for supervised network intrusion detection. *Computers and Security* 8, 459–467.

Melanie, M. (1999). *An Introduction to Genetic Algorithm*. (5th ed.) Bradford Book the MIT press.

Mukkamala, S. & Sung, A. H. (2003). Feature Ranking and selection for intrusion Detection Systems Using Support Vector Machines. *International Conference on Information and Knowledge Engineering*, pp:503-509.

Nguyen, H.A. & Choi, D. (2008). Application of data mining to network intrusion detection: classifier selection model, *Springer-Verlag, Berlin Heidelberg, LNCS*, vol. 5297, pp. 399–408.

Owais, S., Snasel, V., Kromer, P. & Abraham, A. (2008). Survey: Using genetic algorithm approach in intrusion detection systems techniques. *CISIM IEEE*. pp: 76- 81.

Powers, S.T. & He, J. (2008, August 15). A hybrid artificial immune system and self organizing map for network intrusion detection, *Information Sciences* 178, 3024–3042.

Su, MY., Chang, KC., Wei, HF. & Lin, CY. (2008). A real-time network intrusion detection system based on incremental mining approach. *IEEE.* pp: 76- 81.

Tavallaee, M., Bagheri, E., Lu, W. & Ghorbani, A. A. (2009). A Detailed Analysis of the KDD CUP 99 Dataset. Proceedings of the IEEE symposium on computational intelligence in security and defense applications CISDA.

Tavallaee, M., Stakhanova, N. & Ghorbani, A. A. (2010). Towards Credible Evaluation of Anomaly-based Intrusion Detection Methods. *IEEE transactions on systems, Man, and Cybernetics- Part C:* Applications and Reviews, Digital Object Identifier 10.1109/TSMCC.2010.2048428.

Tajbakhsh, A., Rahmati, M. & Mirzaei, A. (2009). Intrusion detection using fuzzy association rules, *Applied Soft Computing* 9, 462–469.

Thomas, C., Sharma, V. & Balakrishnan, N. (2008) Usefulness of DARPA dataset for intrusion detection system evaluation. In: Conference on Data Mining, Intrusion Detection, *Information Assurance and Data Networks Security*, Orlando.

Tsang, C., Kwong, S. & Wang, H. (2007). Genetic-fuzzy rule mining approach and evaluation of feature selection techniques for anomaly intrusion detection. *Pattern Recognition Society*, Elsevier Ltd.

Victoire, T. & Sakthivel, M. (2011). A Refined Differential Evolution Algorithm Based Fuzzy Classifier for Intrusion Detection. *European Journal of Scientific Research*, Vol.65 No.2, pp: 246-259.

Wu,S.X. (2010). Sequential anomaly detection based on temporal-difference learning: principles, models and case studies. *Applied Soft Computing* 10 859–867.

Wu,S.X. & Banzhaf,W. (2012, 1 January). The use of computational intelligence in intrusion detection systems: a review. *Applied Soft Computing Journal* 10 1–35.